Image Classification

👦 Elif 👦 Nikesh 👦 Deepak

What is Image Classification?

- Can you identify the object below?

- Train a system to do the same
- Given image of an object, we would like to train a system to correctly predict what the image represents.

Why Image Classification?

- Facebook's "Alternative text" accessibility feature.
- Google's image search
- Autonomous Cars
- Surveillance/Security

Categories

Image Pre-processing

```
gray_pixels <- function(Image) {
  im <- load.image(Image)
  gray <- grayscale(im)
  resize.im <- resize(gray, w=200, h=200)
  mat <- matrix(resize.im, nrow=1)
  return(mat)
}
```

Data Frame

X1	X2	……..	X40000	CLASS
127	30	76.8	76.7	Tree
234	125	76.9	45.8	Car
45	34.8	35.6	124	Mountain
……. …	……..	……..	…….	……...

200 X 40001

Reduced to:

X1	X2	…	X150	CLASS
45	6	..	76.7	Tree
67	125	..	45.8	Car
7	123	..	124	Mountain
…	...	...	…	...

200 X 151

Our Process is Easy

Algorithms

- Principal Component Analysis
- Support Vector Machine
- K-Nearest Neighbor
- Random Forest

Support Vector Machine

- Use of Support Vector Machine with radial kernel.

- Tuned the SVM using cross validation and the best values for the parameters cost and gamma were 1.265 and 0.001 respectively.

Code Snippet For SVM

```
#SVM with Radial kernel
train <- sample(1:nr, per * nr, rep=F)
test <- setdiff(1:nr, train)
test.df <- prcomp.df[test,]
train.df <- prcomp.df[train,]

tune.svm <-
  tune(svm,as.factor(class) ~.,
       data=train.df,
       kernel="radial",
       scale=F,
       ranges=list(cost=10^seq(-5,5,length=50), gamma=10^seq(-5,5)),
       tunecontrol=tune.control(cross=3))

C <- tune.svm$best.parameters[1]
G <- tune.svm$best.parameters[2]
C
G

svmfit <- svm(as.factor(class) ~.,
              data = train.df,
              kernel = "radial",
              cost = C,
              gamma = G,
              scale = F)
pred.svm <- predict(svmfit, newdata = test.df)
mean(pred.svm != test.df$class)  #Error Rate
table(pred.svm, test.df$class)
```

Confusion Matrix for SVM

Error Rate: 12%

Prediction\ Test	Beach	Car	Mountain	Tree	Waterfall
Beach	11	0	1	0	1
Car	0	7	0	1	0
Mountain	0	0	7	1	0
Tree	0	1	0	8	1
Waterfall	0	0	0	0	10

Random Forest

- Performed comparison of SVM with other algorithms such as Random Forest.

- Random Forest is based on decision trees. This model built 500 random trees and the output was the mean prediction from each of those decision trees.

Code Snippet for Random Forest

```
#Random Forest
rf <- randomForest(as.factor(class)~., data=train.df, ntree=500)
pred <- predict(rf, newdata = test.df)
mean(pred != test.df$class)  #Error rate
table(pred, test.df$class)
```

k-Nearest Neighbor

- An object is classified by a majority vote of its neighbors. It is one of the simplest ML algorithm based on the assumption that an object behaves in similar way to its closest neighbor.

- In k-NN, k refers to the “k” neighbors being considered for the vote.

Code Snippet for k-Nearest Neighbor

```
####K-Nearest-Neighbour Classification

trainsample <- sample(1:nr, per * nr, rep=F)
testsample <- setdiff(1:nr, train)
prcomp_noClass = prcomp.df[,-ncol(prcomp.df)]
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }
knn.df<-as.data.frame(lapply(prcomp_noClass, normalize))
knn_train<-knn.df[trainsample,]
knn_test<-knn.df[testsample,]
knn_train_label<-prcomp.df[train,201]
knn_test_label<-prcomp.df[test,201]
knn_test_pred<-knn(train=knn_train, test=knn_test,cl=knn_train_label,k=10)
mean(knn_test_label != knn_test_pred)
```

Misclassification Error Rates

Support Vector Machine	**12%**
Random Forest	**25%**
K-Nearest Neighbor	**60%**

12%

Whoa! That's close to 0% error!

Thanks!

Any questions?